Setup

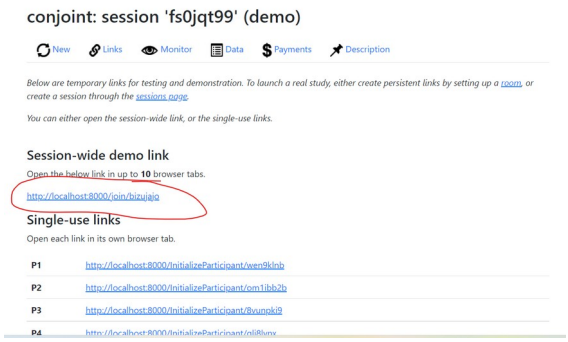
1. Install Selenium by running **pip install selenium** in your terminal.
    - Selenium is the python package that allows us to write functional/acceptance tests through the Selenium WebDriver. This means we can use it to create bots that will give any inputs we want to our survey, so we can test any issues with the survey.
    - Further documentation: selenium · PyPI, 1. Installation — Selenium Python Bindings 2 documentation
2. Install a driver
    - A driver is an .exe that allows to mirror a given browser. I typically use Chrome for testing and this documentation will be aimed at Chrome-based testing, but you can also use Edge, Firefox, or Safari.
    - Download a ChromeDriver here or see this documentation for more information and other browser options
    - Because there are often issues with installing the right version of the driver, I usually try to install it automatically in the code I write, like this:

```
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager

def build_driver():
   # Set up the driver
   return webdriver.Chrome(ChromeDriverManager().install())
```

Writing a bot script

Building the driver & accessing your survey

1. Build the driver. You can do this with my function build_driver or whatever else you have written.
2. Create a session of your survey and copy the session-wide link:
    a. 

3. Then call the driver function .get() to access your survey on the headless browser.

```
driver = build_driver()
driver.get(session_wide_link)
```

4. If you run this code, a browser should appear showing the first page of your survey.

Ways to locate questions & click on answers

- Xpath Method
  - Look at the page that you want to give inputs to. Right-click and then “Inspect” the page. Identify the button that you want by hovering over the html in the right-hand panel. Then right click on the corresponding line of code and click copy > Copy Xpath
    - Obviously this will vary if you’re not using Windows & Chrome, like I am.

  - You can now use this xpath to indicate to the driver that you want to click on that button. See the function below, which clicks “english” and then the “next” button

```
def WelcomePage_v1(driver):
    # click English
    english_xpath = '//*[@id="form"]/div/div/input[2]'
    driver.find_element_by_xpath(english_xpath).click()

    # click "next" button
    next_xpath = '//*[@id="form"]/div/button'
    driver.find_element_by_xpath(next_xpath).click()
```

  - One uses the .click() method if a button on page (e.g a radio button or the next button on the end of the page) should be clicked. If you use a string field, integer field etc. in which you need to write down something you need the .send_keys() method.

```
def Identification(driver):
    # generate random age
    age = random.randint(1, 99)
    # insert into field
    driver.find_element(By.XPATH, '//*[@id="id_age"]').send_keys(age)
```

- While the xpath method is the “easiest” way to locate items on your page, it is not very generalizable and can cause problems if you make changes to your html, like changing the order of questions or the number of inputs.

- Other location methods
  - Other ways to locate a certain element require understanding the basics of how your html works. You can also identify elements with Selenium by name, id, tag name, class name, and css selector.
  - For an example of identifying by name using the same case as the last example, see this function, which also randomizes the input given:

```
import random

def WelcomePage_v2(driver):
   lang_elts = driver.find_elements_by_name('lang')

   # choose a random number between 0 and the number of
elements in lang_elts
   rand_selection = random.randint(0, len(lang_elts)-1)

   # use that random selection to choose which element to
click on
   lang_elts[rand_selection].click()
```

  - Another useful trick is to combine location methods to find certain elements. For example, you might locate a chunk of code contained within a div element and then look for <input> items within that chunk. See this function for an example:

```
def WelcomePage_v3(driver):
   # Identify the <div> container using the xpath
   lang_container_xpath = '//*[@id="form"]/div/div'
   lang_container =
driver.find_element_by_xpath(lang_container_xpath)
```

```
   # Identify the input elements within the div container by
tag name
   lang_elts =
lang_container.find_elements_by_tag_name('input')

   # Randomly select an element to click on
   rand_selection = random.randint(0, len(lang_elts)-1)
   lang_elts[rand_selection].click()
```

Error Handling

Check if an element is visible

- to check whether an element .is_displayed() method is useful. An example how to use this method can be seen below.

```
def check_exists_by_xpath(xpath, driver):
    # use try except so that code will run if element is not displayed
    try:
        # try to find the element
        x = driver.find_element(By.XPATH, xpath)
        # if element is shown return 1
        if x.is_displayed():
            return 1
    except NoSuchElementException:
        # if element is not shown, return 0
        return 0
```